Tree and Graph Mining

Dimitrios Katsaros
Aristotle University, Greece

Yannis Manolopoulos
Aristotle Univeristy, Greece

INTRODUCTION

During the past decade, we have witnessed an explosive growth in our capabilities to both generate and collect data. Various data mining techniques have been proposed and widely employed to discover valid, novel and potentially useful patterns in these data. Data mining involves the discovery of patterns, associations, changes, anomalies, and statistically significant structures and events in huge collections of data.

One of the key success stories of data mining research and practice has been the development of efficient algorithms for discovering frequent itemsets – both sequential (Srikant & Agrawal, 1996) and non-sequential (Agrawal & Srikant, 1994). Generally speaking, these algorithms can extract co-occurrences of items (taking or not taking into account the ordering of items) in an efficient manner. Although the use of sets (or sequences) has effectively modeled many application domains, like market basket analysis, medical records, a lot of applications have emerged whose data models do not fit in the traditional concept of a set (or sequence), but require the deployment of richer abstractions, like graphs or trees. Such graphs or trees arise naturally in a number of different application domains including network intrusion, semantic Web, behavioral modeling, VLSI reverse engineering, link analysis and chemical compound classification.

Thus, the need to extract complex tree-like or graph-like patterns in massive data collections, for instance, in bioinformatics, semistructured or Web databases, became a necessity. The class of exploratory mining tasks, which deal with discovering patterns in massive databases representing complex interactions among entities, is called *Frequent Structure Mining* (FSM) (Zaki, 2002).

In this article we will highlight some strategic application domains where FSM can help provide significant results and subsequently we will survey the most important algorithms that have been proposed for mining graph-like and tree-like substructures in massive data collections.

BACKGROUND

As a motivating example for graph mining consider the problem of mining chemical compounds to discover recurrent (sub) structures. We can model this scenario using a graph for each compound. The vertices of the graphs correspond to different atoms and the graph edges correspond to bonds among the atoms. We can assign a label to each vertex, which corresponds to the atom involved (and maybe to its charge) and a label to each edge, which corresponds to the type of the bond (and maybe to information about the 3D orientation). Once these graphs have been generated, recurrent substructures become frequently occurring subgraphs. These graphs can be used in various tasks, for instance, in classifying chemical compounds (Deshpande, Kuramochi, & Karypis, 2003).

Another application domain where graph mining is of particular interest arises in the field of Web usage analysis (Nanopoulos, Katsaros, & Manolopoulos, 2003). Although various types of usage (traversal) patterns have been proposed to analyze the behavior of a user (Chen, Park, & Yu, 1998), they all have one very significant shortcoming; they are one-dimensional patterns and practically ignore the link structure of the site. In order to perform finer usage analysis, it is possible to look at the entire forward accesses of a user and to mine frequently accessed subgraphs of that site.

Looking for examples where tree mining has been successfully applied, we can find a wealth of them. A characteristic example is XML, which has been a very popular means for representing and storing information of various kinds, because of its modeling flexibility. Since tree-structured XML documents are the most

widely occurring in real applications, one would like to discover the commonly occurring subtrees that appear in the collections. This task could benefit applications, like database caching (Yang, Lee, & Hsu, 2003), storage in relational databases (Deutsch, Fernandez, & Suciu, 1999), building indexes and/or wrappers (Wang & Liu, 2000) and many more.

Tree patterns arise also in bioinformatics. For instance, researchers have collected large amounts of RNA structures, which can be effectively represented using a computer data structure called tree. In order to deduce some information about a newly sequenced RNA, they compare it with known RNA structures, looking for common topological patterns, which provide important insights to the function of the RNA (Shapiro & Zhang, 1990). Another application of tree mining in bioinformatics is found in the context of constructing phylogenetic trees (Shasha, Wang, & Zhang, 2004), where the task of phylogeny reconstruction algorithms it to use biological information about a set of e.g., taxa, in order to reconstruct an ancestral history linking together all the taxa in the set.

There are two distinct formulations for the problem of mining frequent graph (tree) substructures and are referred to as the *graph-transaction* (*tree-transaction*) setting and the *single-graph* (*single-tree*) setting. In the graph-transaction setting, the input to the pattern-mining algorithm is a set of relatively small graphs (called transactions), whereas in the single-graph setting the input data is a single large graph. The difference affects the way the frequency of the various patterns is determined. For the former, the frequency of a pattern is determined by the number of graph transactions that the pattern occurs in, irrespective of how many times a pattern occurs in a particular transaction, whereas in the latter, the frequency of a pattern is based on the number of its occurrences (i.e., embeddings) in the single graph. The algorithms developed for the graph-transaction setting can be modified to solve the single-graph setting, and vice-versa.

Depending also on the application domain, the considered graphs (trees) can be ordered or unordered, directed or undirected. No matter what these characteristics are, the (sub)graph mining problem can be defined as follows. (A similar definition can be given for the tree mining problem.) Given as input a database of graphs and a user-defined real number $0<\sigma\leq 1$, we need to find all frequent subgraphs, where the word "frequent" implies those subgraphs with frequency larger than or equal to the threshold σ. (In the following, equivalently to the term *frequency*, we use the term *support.*) We illustrate this problem for the case of graph-transaction, labeled, undirected graphs, with $\sigma=2/3$. The input and output of such an algorithm are given in Figure 1.

Although the very first attempts to deal with the problem of discovering substructure patterns from massive graph or tree data are dated back to the early 90's (Cook & Holder, 1994), only recently the field of mining for graph and tree patterns has flourished. A wealth of algorithms has been proposed, most of which are based on the original level-wise *Apriori* algorithm for mining frequent itemsets (Agrawal & Srikant, 1994). Next, we will survey the most important of them.

ALGORITHMS FOR GRAPH MINING

The graph is one of the most fundamental constructions studied in mathematics and thus, numerous classes of substructures are targeted by graph mining. These substructures include the *generic subgraph*, *induced subgraph*, *connected subgraph*, (ordered and unordered) *tree* and *path* (see Figure 2). We give the definitions of these substructures in the next paragraph and subsequently present the graph mining algorithms, able to discover all frequent substructures of any kind mentioned earlier.

Following mathematical terminology, a graph is represented as G(V, E, f), where V is a set of vertices, E is a set of edges connecting pairs of vertices and f is a function $f:E\rightarrow VxV$. For instance, in Figure 2 we see that $f(e_1)=(v_1,v_2)$. We say that *GS*(Vs,Es,f) is a *generic subgraph* of G, if $Vs\subset V$, $Es\subset E$ and $v_i,v_j\in Vs$ for all edges $f(e_k)=(v_i,v_j)\in Es$. An *induced subgraph ISG*(Vs,Es,f) of G has a subset of vertices of G and the same edges between pairs of vertices as in G, in other words, $Vs\subset V$, $Es\subset E$ and $\forall v_i,v_j\in Vs$, $e_k=(v_i,v_j)\in Es\Leftrightarrow f(e_k)=(v_i,v_j)\in E$. We say that *CSG*(Vs,Es,f) is a connected subgraph of G, if $Vs\subset V$, $Es\subset E$ and all vertices in Vs are reachable through some edges in Es. An acyclic subgraph of G is called a *tree T*. Finally, a tree of G which does not include any braches is a *path P* in G.

The first algorithm for mining all frequent subgraph patterns is *AGM* (Inocuchi, Washio, & Motoda, 2000, 2003). AGM can mine various types of patterns, namely generic subgraphs, induced subgraphs, connected subgraphs, ordered and unordered trees and subpaths.